Preimage Attacks on 3, 4, and 5-pass HAVAL

Yu Sasaki and Kazumaro Aoki

NTT, 3-9-11 Midoricho, Musashino-shi, Tokyo, 180-8585 Japan

Abstract. This paper proposes preimage attacks on hash function HAVAL whose output length is 256 bits. This paper has three main contributions; a preimage attack on 3-pass HAVAL at the complexity of 2^{225}, a preimage attack on 4-pass HAVAL at the complexity of 2^{241}, and a preimage attack on 5-pass HAVAL reduced to 151 steps at the complexity of 2^{241}. Moreover, we optimize the computational order for brute-force attack on full 5-pass HAVAL and its complexity is $2^{254.89}$. As far as we know, the proposed attack on 3-pass HAVAL is the best attack and there is no preimage attack so far on 4-pass and 5-pass HAVAL. Note that the complexity of the previous best attack on 3-pass HAVAL is 2^{230}. Technically, our attacks find pseudo-preimages of HAVAL by combining the meet-in-the-middle and local-collision approaches, then convert pseudo-preimages to a preimage by using a generic algorithm.

keywords: HAVAL, splice-and-cut, meet-in-the-middle, local collision, hash function, one-way, preimage

1 Introduction

Cryptographic hash functions are important primitives to build secure schemes. A hash function takes arbitrarily long bit string and outputs a hash value with a fixed length. A hash function is required to satisfy the security properties such as collision resistance, 2nd preimage resistance, and preimage resistance. When the length of the hash value is n bits, a collision, a 2nd preimage, and a preimage should not be computed faster than $2^{n/2}$, 2^n, and 2^n operations, respectively.

HAVAL [18] is one of the dedicated hash functions and has relatively long history. HAVAL is based on Merkle-Damgård construction, and its compression function is similar to MD5 [10]. The basic operation of HAVAL is done in 32 bits that is the same as MD5. Therefore, 32-bit values are called *words*. However, the interface of the HAVAL compression function is doubled compared to MD5, that is, the number of chaining variables and the message length of the compression function are 8 words and 32 words respectively. The nonlinear function of HAVAL takes seven words as input and outputs a word. So, one step of HAVAL only changes one word out of 8 words of the internal state. To satisfy several security requirements, HAVAL has three variants called x-pass HAVAL ($x = 3, 4, 5$). x-pass HAVAL consists of $32x$ steps.

Due to the simple structure, there are several cryptoanalytic results on HAVAL as shown in the next paragraph. However, regarding the preimage attack, there is only one result on 3-pass HAVAL [2]. In this paper, we propose preimage attacks on HAVAL: the best attack on 3-pass HAVAL so far, the first attack on 4-pass HAVAL and 5-pass HAVAL.

Table 1. Comparison of preimage attacks on HAVAL

Attack target	Number of steps	Attack type	Previous work [2]	**Our attack**	
				strategy 1*	strategy 2
3-pass	96 (Full)	Pseudo-preimage	2^{224}	2^{253}	2^{192}
		Preimage	2^{230}		2^{225}
4-pass	128 (Full)	Pseudo-preimage	-	$2^{254.43}$	2^{224}
		Preimage	-		2^{241}
5-pass	151 (Steps 0-150)	Pseudo-preimage	-	Not evaluated	2^{224}
		Preimage	-		2^{241}
	160 (Full)	Pseudo-preimage	-	$2^{254.89}$	$2^{253.81}$∗
		Preimage	-		—

∗ This attack is a kind of brute force attack, but the computation is optimized.

Known previous results except for the preimage attack are as follows: collision attacks on 3-pass HAVAL are discussed in Ref. [12, 11, 13, 14], and those on 4-pass HAVAL are discussed in Ref. [15, 17]. Note that a real collision has been generated up to 4-pass HAVAL. Theoretically, a collision of 5-pass HAVAL can be generated in 2^{123} compression function evaluations [17] that is faster than the birthday paradox for 256-bit output. (Hereafter, we omit the unit of complexity whenever it is obvious and it is the number of "compression function evaluation.") Non-randomness of 4-pass and 5-pass of HAVAL in the encryption mode is analyzed by Ref. [6, 16]. The security of the HMAC-HAVAL is analyzed by Ref. [5]. A 2nd preimage attack on 3-pass HAVAL and its application to HMAC-3-pass HAVAL are proposed by Ref. [7]. However, this 2nd preimage attack is different from the one usually considered. In Ref. [8] a useful statement to clarify the difference of these two types of 2nd preimage attacks is shown. The attack of Ref. [7] can generate a 2nd preimage at the complexity of one compression function with a probability of 2^{-114} for a given random message, and it requires the complexity of 2^{128} with a probability of $1 - 2^{-114}$. Therefore, the average complexity is very close to 2^{128}. Consequently, no result that produces a 2nd preimage of any given message is known. Moreover, no result is known on preimage attack on HAVAL, except for the recent result on 3-pass HAVAL [2].

1.1 Related work regarding preimage attack

In 2008, a preimage attack on MD4 was proposed by Leurent [8]. The attack first generates pseudo-preimages based on the Dobbertin's pioneering work [4], and converts a pseudo-preimage attack to a preimage attack by using the generic approach [9, Fact9.99][1]. Preimage attacks on step-reduced MD5 and full 3-pass HAVAL are proposed by Aumasson et al. [2], whose approach is based on the meet-in-the-middle technique. Preimage attacks on full MD4 and 63-step MD5 are proposed by [1], whose approach is also based on the meet-in-the-middle technique. Note, both of [2, 1] use the conversion algorithm of [9, Fact9.99].

[1] The following works that compute a preimage from partial-pseudo-preimages also use this kind of conversion. The method of the conversion from partial-pseudo-preimages to a preimage is improved by using hash-tree [8] and P^{3}graph [3].

In the meet-in-the-middle attack of Aumasson et al. [2], a compression function is divided into the first half and the last half, and both computation results are compared in the middle. They also use new techniques that make the attack efficient by using the absorption properties of Boolean functions. On the other hand, Aoki and Sasaki propose new techniques to apply the meet-in-the-middle attack to not only the first half and the last half but also any two consecutive parts of a compression function [1]. This paper combines the techniques of Ref. [1, 2], and attacks more passes of HAVAL.

1.2 Our contributions

In this paper, we propose preimage attacks on 3-, 4-, and 5-pass HAVAL whose output length is 256 bits. First, we consider a strategy to find preimages of 3-, 4-, and 5-pass HAVAL faster than the brute force attack by a few bits (strategy 1). Second, we consider another strategy that can find a preimages of 3-, 4-, and 5-pass HAVAL much more efficiently by combining techniques of [1] and [2] (strategy 2). As a result of applying strategy 2 to each pass of HAVAL, we find the best preimage attack so far on 3-pass HAVAL by using the techniques of [1], the first preimage attack on 4-pass HAVAL by combining techniques of [1] and [2], and the first preimage attack on step-reduced 5-pass HAVAL by combining techniques of [1] and [2] and further improving a technique of [2]. We summarize the results of the previous work and ours in Table 1.

Organization of this paper is as follows. Section 2 introduces the specification of HAVAL and techniques of existing attacks. Section 3 gives two strategies of the preimage attack that can be applied to HAVAL and *other hash functions* whose message expansion is similar to HAVAL. Regarding the technique in Ref. [2] as an application of a local collision, we can compute preimages of a hash function that has more rounds. Section 4 describes attacks on HAVAL following the strategy 1. Section 5 describes attacks on HAVAL following the strategy 2. Finally, we conclude this paper in Section 6.

2 Previous works: specification and techniques for preimage attacks

2.1 Description of HAVAL

HAVAL is a hash function proposed by Zheng et al. in 1992, which compresses a message up to $(2^{64} - 1)$ bits into either 128, 160, 192, 224, or 256 bits. Since this paper only analyzes 256-bit version, we only describe the specification for 256 bits. HAVAL iteratively computes a step function to compute a hash value. The number of steps is chosen from either 96, 128, or 160, where corresponding HAVAL algorithms are called 3-pass HAVAL, 4-pass HAVAL, and 5-pass HAVAL, respectively. HAVAL has the Merkle-Damgård structure, which uses 256-bit (8-word) chaining variables and a 1024-bit (32-word) message block to compute a compression function.

An input message M is processed to be a multiple of 1024 bits by the padding procedure. A single bit '1' is appended followed by '0's until the length becomes 944 modulo 1024. At the last, 3-bit field representing a version number

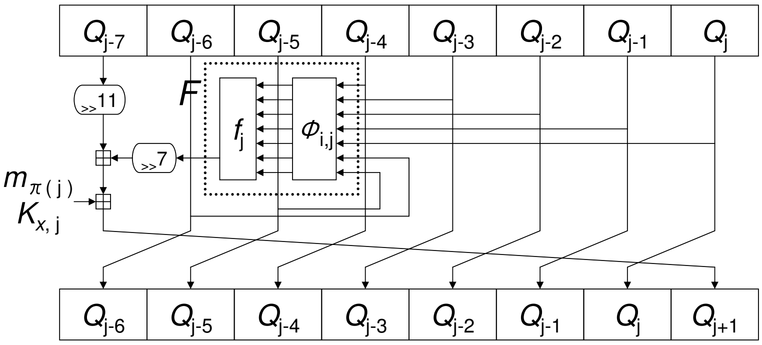


Fig. 1. Step function of HAVAL

Table 2. HAVAL message expansion

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
5	14	26	18	11	28	7	16	0	23	20	22	1	10	4	8	30	3	21	9	17	24	29	6	19	12	15	13	2	25	31	27
19	9	4	20	28	17	8	22	29	14	25	12	24	30	16	26	31	15	7	3	1	0	18	27	13	6	21	10	23	11	5	2
24	4	0	14	2	7	28	23	26	6	30	20	18	25	19	3	22	11	31	21	8	27	12	9	1	29	5	15	17	10	16	13
27	3	21	26	17	11	20	29	19	0	12	7	13	8	31	10	5	9	14	30	18	6	28	24	2	23	16	22	4	1	25	15

of HAVAL, 3-bit field representing the number of the pass used, 10-bit field representing the output length, and 64-bit field representing an unpadded message length are appended.

Padded message M^* is separated into 1024-bit message blocks $(M_0, M_1, \ldots, M_{n-1})$. Let $CF : \{0,1\}^{256} \times \{0,1\}^{1024} \rightarrow \{0,1\}^{256}$ be the compression function of HAVAL. A hash value is computed as follows.

1. $H_0 \leftarrow IV$,
2. $H_{i+1} \leftarrow CF(H_i, M_i)$ for $i = 0, 1, \ldots, n-1$,

where H_i is a 256-bit value and IV is the initial value defined in the specification. Finally, H_n is output as a hash value of M.

Compression Function. The compression function $H_{i+1} \leftarrow CF(H_i, M_i)$ is computed as follows.

1. M_i is divided into 32-bit message words m_j $(j = 0, 1, \ldots, 31)$.
2. $p_0 \leftarrow H_i$.
3. $p_{j+1} \leftarrow R_j(p_j, m_{\pi(j)})$ for $j = 0, 1, \ldots, k$, where $k = 32x - 1$ for x-pass.
4. Output $H_{i+1} (= p_k + H_i)$, where "+" denotes a 32-bit word-wise addition. In this paper, we similarly use "$-$" to denote a 32-bit word-wise subtraction.

R_j is the step function for Step j. Let Q_j be a 32-bit value that satisfies $p_j = (Q_{j-7} \| Q_{j-6} \| Q_{j-5} \| Q_{j-4} \| Q_{j-3} \| Q_{j-2} \| Q_{j-1} \| Q_j)$. R_j for x-pass HAVAL $(x \in \{3,4,5\})$ is defined as follows:

$$\begin{cases} T = f_j \circ \phi_{x,j}(Q_{j-6}, Q_{j-5}, Q_{j-4}, Q_{j-3}, Q_{j-2}, Q_{j-1}, Q_j) \\ R_j(p_j, m_{\pi(j)}) = (Q_{j-7} \ggg 11) + (T \ggg 7) + m_{\pi(j)} + K_{x,j} \end{cases}$$

$$
\begin{aligned}
0 \leq j \leq 31 : f_j(x_6, x_5, \ldots, x_0) &= x_1x_4 \oplus x_2x_5 \oplus x_3x_6 \oplus x_0x_1 \oplus x_0 \\
32 \leq j \leq 63 : f_j(x_6, x_5, \ldots, x_0) &= x_1x_2x_3 \oplus x_2x_4x_5 \oplus x_1x_2 \oplus x_1x_4 \oplus \\
&\quad x_2x_6 \oplus x_3x_5 \oplus x_4x_5 \oplus x_0x_2 \oplus x_0 \\
64 \leq j \leq 95 : f_j(x_6, x_5, \ldots, x_0) &= x_1x_2x_3 \oplus x_1x_4 \oplus x_2x_5 \oplus x_3x_6 \oplus x_0x_3 \oplus x_0 \\
96 \leq j \leq 127 : f_j(x_6, x_5, \ldots, x_0) &= x_1x_2x_3 \oplus x_2x_4x_5 \oplus x_3x_4x_6 \oplus \\
&\quad x_1x_4 \oplus x_2x_6 \oplus x_3x_4 \oplus x_3x_5 \oplus \\
&\quad x_3x_6 \oplus x_4x_5 \oplus x_4x_6 \oplus x_0x_4 \oplus x_0 \\
128 \leq j \leq 159 : f_j(x_6, x_5, \ldots, x_0) &= x_1x_4 \oplus x_2x_5 \oplus x_3x_6 \oplus x_0x_1x_2x_3 \oplus x_0x_5 \oplus x_0
\end{aligned}
$$

x_ax_b represents bitwise AND operation.

Fig. 2. Boolean Functions of HAVAL

Table 3. Wordwise rotation of HAVAL

	$x_6\ x_5\ x_4\ x_3\ x_2\ x_1\ x_0$ $\downarrow\ \downarrow\ \downarrow\ \downarrow\ \downarrow\ \downarrow\ \downarrow$		$x_6\ x_5\ x_4\ x_3\ x_2\ x_1\ x_0$ $\downarrow\ \downarrow\ \downarrow\ \downarrow\ \downarrow\ \downarrow\ \downarrow$		$x_6\ x_5\ x_4\ x_3\ x_2\ x_1\ x_0$ $\downarrow\ \downarrow\ \downarrow\ \downarrow\ \downarrow\ \downarrow\ \downarrow$
$\phi_{3,1}$	$x_1\ x_0\ x_3\ x_5\ x_6\ x_2\ x_4$	$\phi_{4,1}$	$x_2\ x_6\ x_1\ x_4\ x_5\ x_3\ x_0$	$\phi_{5,1}$	$x_3\ x_4\ x_1\ x_0\ x_5\ x_2\ x_6$
$\phi_{3,2}$	$x_4\ x_2\ x_1\ x_0\ x_5\ x_3\ x_6$	$\phi_{4,2}$	$x_3\ x_5\ x_2\ x_0\ x_1\ x_6\ x_4$	$\phi_{5,2}$	$x_6\ x_2\ x_1\ x_0\ x_3\ x_4\ x_5$
$\phi_{3,3}$	$x_6\ x_1\ x_2\ x_3\ x_4\ x_5\ x_0$	$\phi_{4,3}$	$x_1\ x_4\ x_3\ x_6\ x_0\ x_2\ x_5$	$\phi_{5,3}$	$x_2\ x_6\ x_0\ x_4\ x_3\ x_1\ x_5$
-	-	$\phi_{4,4}$	$x_6\ x_4\ x_0\ x_5\ x_2\ x_1\ x_3$	$\phi_{5,4}$	$x_1\ x_5\ x_3\ x_2\ x_0\ x_4\ x_6$
-	-	-	-	$\phi_{5,5}$	$x_2\ x_5\ x_0\ x_6\ x_4\ x_3\ x_1$

where f_j is a bitwise Boolean function defined in Fig. 2, $\phi_{i,j}$ is a wordwise permutation defined in Table 3, π_j is a message expansion function defined in Table 2, $\ggg n$ is n-bit right rotation, and $K_{x,j}$ is a constant defined in the specification. We show a graph of the step function in Fig. 1. Note that $R_j^{-1}(\cdot, m_{\pi(j)})$ can be computed in almost the same complexity as that of R_j.

2.2 Converting pseudo-preimages to a preimage

For a given hash value y, a pseudo-preimage is a pair of (x, M) such that $CF(x, M) = y$, where x may not equal to IV and CF is a compression function of a Merkle-Damgård hash function. There is a generic algorithm that converts a pseudo-preimage attack to a preimage attack [9, Fact9.99]. Let the complexity of a pseudo-preimage attack be 2^k. The procedure of this attack when the hash value is n-bit long is as follows.

1. Generate $2^{(n-k)/2}$ pseudo-preimages at the complexity of $2^k \cdot 2^{(n-k)/2}$.
2. Generate $2^{(n+k)/2}$ messages that start from the IV, and compute their hash values.

One of these hash values is expected to be matched. The complexity of this attack is $2^k \cdot 2^{(n-k)/2} + 2^{(n+k)/2} = 2^{1+(n+k)/2}$.

This algorithm has been used in previous preimage attacks [8, 2, 1].

2.3 Preimage attacks on 3-pass HAVAL

Aumasson et al. proposed two attacks that find a preimage of 3-pass HAVAL at the complexity of 2^{230}, and the attacks require 16×2^{64} words of memory [2].

Both attacks take an approach of the meet-in-the-middle attack. In this paper, we are particularly interested in the Attack A of their paper [2, Algorithm 4].

In the Attack A of [2, Algorithm 4], the authors focused attention on the location of the message words m_5 and m_6, where m_5 appears at Step 5, 32, and 94 and m_6 appears at Step 6, 55, and 89 as shown in Table 2[2]. First, chaining variables p_0 to p_6, where p_0 is IV and p_{i+1} is the 256-bit output of the i-th step, are fixed so that the change of m_6 in Step 6 is guaranteed to be absorbed by changing Q_{-1}, which is the seventh word of the IV. Similarly, chaining variables p_{95} and p_{96} are fixed so that the change of m_5 in Step 94 is guaranteed to be absorbed by changing Q_{95}, which is the seventh word of p_{96}. Due to this effort, computation for Step 0 to 47 becomes independent of m_6, and computation for Step 95 to 48 becomes independent of m_5. The authors of [2] and we call these independent words *neutral words.*

Finally, the authors apply the meet-in-the-middle attack to find a pseudo-preimage of a given hash value $H_n = (H^a \| H^b \| H^c \| H^d \| H^e \| H^f \| H^g \| H^h)$. The rough sketch of the procedure is as follows. Refer to [2] for details.

Algorithm

1. Fix $m_x, x \notin \{5, 6\}$ and $p_y, y \in \{0, \ldots, 6, 95, 96\}$ so that changes of m_6 in Step 6 and of m_5 in Step 94 are absorbed and $p_0 + p_{96} = H_n$ is satisfied except for $Q_{-1} + Q_{95} = H^g$.
2. For all 64 bits of (m_5, Q_{-1}), compute $p_{j+1} \leftarrow R_j(p_j, m_{\pi(j)})$ for $j = 0, 1, \ldots, 47$, and store them in a table.
3. For all 64 bits of (m_6, Q_{95}), compute $p_j \leftarrow R_j^{-1}(p_{j+1}, m_{\pi(j)})$ for $j = 95, 94, \ldots, 48$. Then, check if resulting p_{48} are matched with p_{48}s in the table.
4. For all matched pairs, check if $Q_{-1} + Q_{95} = H^g$ is satisfied.

In the above procedure, the meet-in-the-middle attack saves the complexity of 64 bits but step 4 of the procedure succeeds with a probability of 2^{-32}. Thus, this attack is faster than the brute force attack by the factor of 2^{32}.

2.4 Preimage attacks on MD4 and MD5

Preimage attacks on MD4 and MD5 are proposed by Aoki and Sasaki [1]. They proposed a new technique called the splice-and-cut technique.
Splice-and-Cut: Splice the last and the first step and divide the attack target into two chunks of steps so that each chunk includes at least one message word that is independent of the other chunk. Then, pseudo-preimages are computed by the meet-in-the-middle approach.

Different from Aumasson et al. [2], Aoki and Sasaki focused attention on the property that chaining variables in the first and last steps can be considered to be consecutive by the equation $p_0 = H_n - p_{\text{last}}$. This idea enables them to start the meet-in-the-middle attack from *any* step.

Aoki and Sasaki also proposes another technique named *partial matching.* This technique enables attackers to skip several steps when they search for good chunks in the attack target. Assume that one of divided chunks provides the value

[2] We number the first step as 0.

of p_i, where $p_i = (Q_{i-7} \| Q_{i-6} \| Q_{i-5} \| Q_{i-4} \| Q_{i-3} \| Q_{i-2} \| Q_{i-1} \| Q_i)$ and the other chunk provides the value of p_{i-4}, where $p_{i-4} = (Q_{i-11} \| Q_{i-10} \| Q_{i-9} \| Q_{i-8} \| Q_{i-7} \| Q_{i-6} \| Q_{i-5} \| Q_{i-4})$. p_i and p_{i-4} cannot be directly compared, however, a part of these values, that is, $Q_{i-7}, Q_{i-6}, Q_{i-5}$, and Q_{i-4} can be compared immediately. In such a case, we can ignore the value of $m_{\pi(i-1)}, m_{\pi(i-2)}, m_{\pi(i-3)}$, and $m_{\pi(i-4)}$ when we perform the meet-in-the-middle attack.

3 General strategies of our preimage attack

3.1 Strategy 1: speed up the brute-force attack

This is a technique that enables us to quickly search for a message which connects a given initial value IV and a given hash value H_n. The idea is to reuse an intermediate value of computation of a message when we compute different messages. Assume m_a and m_b form a local collision in the first round, that is, any change of m_a can be offset by changing m_b, and these messages appear at Steps $s_1, s_2, (s_1 < s_2)$ in the second round. In this case, the computation result until Step s_1 can be reused with all m_a and corresponding m_b.

Moreover, since IV and H_n are fixed, the values of chaining variables in the last round can also be reused. Let steps at which m_a and m_b are used be $s_3, s_4, (s_3 < s_4)$. In this case, the computation result from Step s_4 to the last can be reused.

Notice, this technique can also be achieved by inserting local collision in the last round.

3.2 Strategy 2: finding pseudo-preimages by the meet-in-the-middle attack

Combining the splice-and-cut and local-collision. The technique proposed by Aumasson et al. [2] is for finding a pseudo-preimage by applying the meet-in-the-middle attack that starts from the first step and the last step. On the other hand, the splice-and-cut and partial-matching techniques proposed by Aoki and Sasaki [1] are for finding a pseudo-preimage by applying the meet-in-the-middle attack that starts from an intermediate step. We found that these two techniques can be combined together, and more steps might be attacked.

Aumasson et al. use the fact that m_6 is used near the first step, m_5 is used near the last step, and corresponding chaining variables appear in the same equation for the computing hash value. We found that their technique can be used at not only the first and last several steps but also intermediate steps.

Observation: The key idea of the attack is searching for message words that can form a local collision. In fact, their selection of message words can be considered as a local collision that starts with Step 94 and ends with Step 6.

The graphical explanation is shown in Fig. 3. Cells denote 32-bit chaining variables and highlighted cells denote chaining variables whose values are changed according to the selection of values of neutral words (m_5, m_6). The left diagram explains the attack procedure of Aumasson et al., and the right diagram describes it in a different step order to show (m_5, m_6) forms a local

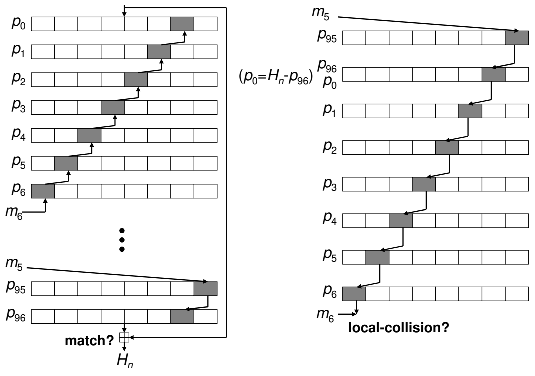


Fig. 3. A local collision formed by the neutral words used by Aumasson et al. [2]

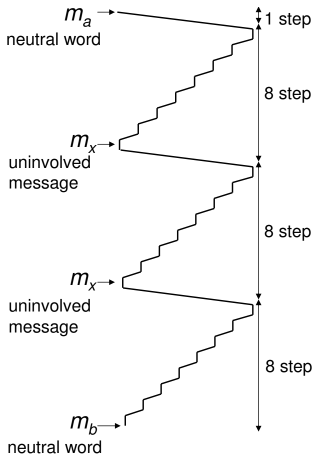


Fig. 4. A long collision pass used in the splice-and-cut technique

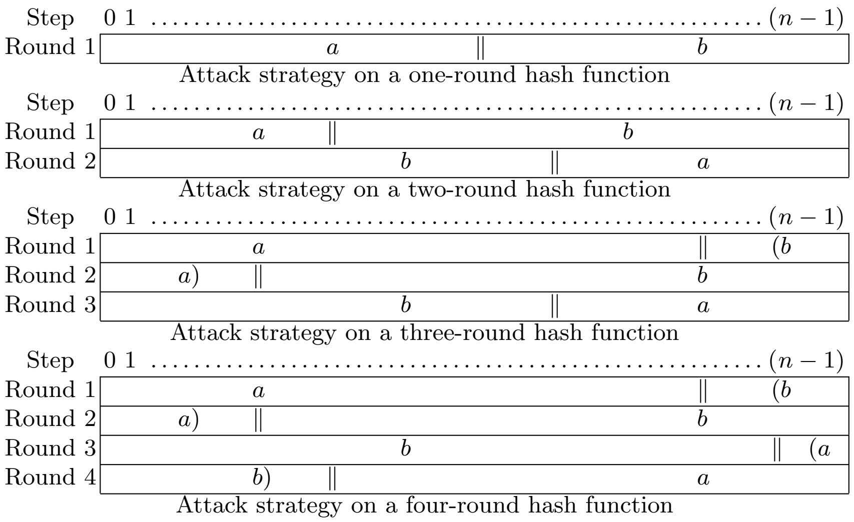
collision. Note, in the splice-and-cut technique, the first and last steps are considered to be consecutive by the equation $p_0 = H_n - p_{96}$, which can be ignored when we analyze the dependency of message words.

As you can see in Fig. 3, the technique of Aumasson et al. [2] can be inserted in any part of an attack target. Therefore, this can be combined with the splice-and-cut technique. For convenience, we call this technique *local-collision technique*, and we summarize the property of the local-collision technique.

New technique 1. Local-Collision: When we search for chunks in an attack target, neutral words forming a local collision can be ignored. This occurs when neutral words appear $(L+1)$ *steps away and other chaining variables can be guaranteed not to be affected by the local collision, where L represents the number of chaining variables (e.g.* $L = 4$ *for* MD5*,* $L = 8$ *for* HAVAL*).*

Extension to use long collision paths. The local-collision technique described above can be extended to use a long collision path as shown in Fig. 4.

In HAVAL, the influence of changing $m_{\pi(i)}$ can be offset by changing $m_{\pi(i+8n)}$, $n \geq 1$. In this case, $m_{\pi(i+8k)}$, $1 \leq k < n$ can be any message word. We call $m_{\pi(i+8k)}$ *uninvolved messages.* As long as the meet-in-the-middle attack with a local collision such as the attack approach of Aumasson et al. is taken, neutral words can also be used as uninvolved messages. On the other hand, in our approach explained in Section 5.3, we use "meet-in-the-middle attack" which uses two tables but does not get the gain of the time-to-memory conversion. Thus, neutral words require to increase the complexity of about $n/$(number of all steps), since we need to fix all variables within local collision steps before we perform the "meet-in-the-middle attack". We also note that the changes of a 32-bit chaining variable corresponding to neutral words must be absorbed in the Boolean functions so that other chaining variables are not changed. Achieving this tends to



Attack strategy on a one-round hash function

Attack strategy on a two-round hash function

Attack strategy on a three-round hash function

Attack strategy on a four-round hash function

Fig. 5. Attack strategies on a hash function with up to 4 rounds

be difficult if several message words appear twice or messages used as padding string appear in a local collision path.

Number of rounds that can be attacked. The meet-in-the-middle attack works very efficiently if the message expansion consists of a permutation of message word order in each round like MD5 or HAVAL. In this section, we formalize how many rounds can be attacked. Attack strategies are also drawn in Fig. 5.

We explain how to attack a hash function that has only one-round. Let us divide the attack target into the first half and the last half steps. In a round, each message appears only once. Therefore, any pair of message words used in the first and second chunks are independent each other, hence they can be used as the neutral words. Finally, we perform the meet-in-the-middle attack between the first chunk including m_a and the second chunk including m_b.

To attack a two-round hash function, we use the property that chaining variables in the first and last steps can be considered to be consecutive. Let a pair of message words (m_a, m_b) appear in the first round in this order. In the second round, if m_b is used in an earlier step than m_a, the attack target can be divided into two chunks so that one chunk includes a neutral word m_a and the other chunk includes m_b. Therefore, a pseudo-preimage attack can be achieved by the splice-and-cut technique.

A three-round hash function can be attacked by combining the splice-and-cut technique and one of the partial-matching or local collision techniques. Assume (m_a, m_b) is a pair of message words that can be skipped by using the partial-matching or local-collision technique. In Fig. 5, skipped steps are indicated by parentheses. If the same strategy for the two-round attack can be applied in the rest of steps, a pseudo-preimage attack can be achieved.

To attack a four-round hash function, we need to use all techniques. At the beginning of two chunks, we skip several steps by the local-collision technique,

Table 4. Message word distribution for fast brute-force attack on 3-pass HAVAL

Step	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	$\cdots$	29	30	31
index	0	1	2	3	4	5	6	7	8	⑨	10	11	12	13	14	15	16	⑰	18	19	20	21	$\cdots$	29	30	31
	reused									local collision									reused							
Step	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	$\cdots$	61	62	63
index	5	14	26	18	11	28	7	16	0	23	20	22	1	10	4	8	30	3	21	⑨	⑰	24	$\cdots$	25	31	27
	reused																									
Step	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	$\cdots$	93	94	95
index	19	⑨	4	20	28	⑰	8	22	29	14	25	12	24	30	16	26	31	15	7	3	1	0	$\cdots$	11	5	2
							reused																			

and at the end of two chunks, we skip several steps by the partial-matching technique. Both skipped steps need to include both neutral words.

4 Preimage attacks on HAVAL following the strategy 1

We apply the general strategy 1 explained in Section 3 to all passes of HAVAL. The memory requirement of the attack is negligible.

First, we consider a preimage attack on 3-pass HAVAL. According to the message expansion of HAVAL shown in Table 2, if we make a local collision from Steps 9 to 17, computation results for 77 steps out of 96 steps can be reused among different messages. The message word distribution for this attack is shown in Table 4.

The attack procedure is as follows:

Attack procedure

1. Fix m_{29}, m_{30}, and m_{31} to satisfy the padding for a 1-block message.
2. Temporarily determine m_9 and m_{17}, and determine chaining variables and messages $m_i, i \notin \{9, 17, 29, 30, 31\}$ so that Steps 9-17 form a local collision[3].
3. Randomly determine other message words that are not specified yet.
4. Compute $R_j(p_j, m_{\pi(j)})$ for $j = 0, 1, \ldots, 50$ and compute $R_j^{-1}(p_{j+1}, m_{\pi(j)})$ for $j = 95, 94, \ldots, 70$, where $p_{96} = H_n - IV$. Store the values of p_{51} and p_{70} in a table, where $p_{70} = (Q_{63} \| Q_{64} \| Q_{65} \| Q_{66} \| Q_{67} \| Q_{68} \| Q_{69} \| Q_{70})$.
5. For all 32 bits of m_9, compute m_{17} so that the value of Q_{18} does not change. Then, compute $R_j(p_j, m_{\pi(j)})$ for $j = 51, 52, \ldots, 62$ and check whether computed Q_{63} is in the table or not. If it is in the table, compute Q_{63}, …, Q_{70} and check all values are matched. Otherwise, choose other m_9 and repeat this process.

The complexity of the above procedure is $2^{29} (= 2^{32} \cdot \frac{12}{96})$ and success probability of step 5 is $2^{-224} (= 2^{-256} \cdot 2^{32})$. Therefore, by repeating the procedure 2^{224} times by changing the values of $m_i, 18 \leq i \leq 28$, a message that connects a given IV and H_n will be found at the complexity of $2^{253} (= 2^{29} \cdot 2^{224})$.

On 4-pass HAVAL, the attack procedure is similar to 3-pass HAVAL. Applying local collision in the last round between Steps 102-110, the complexity

[3] How to determine the chaining variables and messages to obtain a local collision is explained in Section 5.2. A local collision for this attack can be obtained in the similar method.

Table 5. Message word distribution for 3-pass HAVAL

Step	0	1	2	3	4	5	6	7	8	9	10	11	12	13	···	21	22	23	24	25	26	27	28	29	30	31
index	⓪	①	2	3	4	⑤	6	7	8	9	10	⑪	12	13	···	21	22	23	24	25	26	27	28	29	30	31
	skip		first chunk																							
Step	32	33	34	35	36	37	38	39	40	41	42	43	44	45	···	53	54	55	56	57	58	59	60	61	62	63
index	⑤	14	26	18	⑪	28	7	16	⓪	23	20	22	①	10	···	24	29	6	19	12	15	13	2	25	31	27
	first chunk								second chunk																	
Step	64	65	66	67	68	69	70	71	72	73	74	75	···	83	84	85	86	87	88	89	90	91	92	93	94	95
index	19	9	4	20	28	17	8	22	29	14	25	12	···	3	①	⓪	18	27	13	6	21	10	23	⑪	⑤	2
	second chunk																							skip		

of the attack is $2^{256} \cdot \frac{128-(19+59+7)}{128} \approx 2^{254.43}$. On 5-pass HAVAL, applying local collision in the first round between Steps 19-27, the complexity of the attack is $2^{256} \cdot \frac{160-(56+23+7)}{160} \approx 2^{254.89}$.

5 Preimage attacks on HAVAL following the strategy 2

Our general strategy 1 can work for all passes of HAVAL, however, the efficiency is not so high. This section further reduces the complexity of preimage attacks by using the general strategy 2, which uses the meet-in-the-middle approach.

5.1 A preimage attack on 3-pass HAVAL

We propose a preimage attack on 3-pass HAVAL, which finds a pseudo-preimage of 3-pass HAVAL at the complexity of 2^{192}, and is converted to a preimage attack of the complexity of 2^{225}. Thus, the resulting preimage is 2-block long. This attack uses the splice-and-cut and partial-matching techniques as shown in Table 5.

The attack procedure for a hash value $H_n = (H^a \| H^b \| H^c \| H^d \| H^e \| H^f \| H^g \| H^h)$ is as follows.

Attack procedure

1. Fix m_{29}, m_{30}, and m_{31} to satisfy the padding for a 2-block message.
2. Fix m_i ($i \notin \{0, 1, 5, 11\}$) and p_{40} to randomly chosen values.
3. For all (m_0, m_1), do: $p_{j+1} \leftarrow R_j(p_j, m_{\pi(j)})$ for $j = 40, 41, \ldots, 92$.
4. Make a table of $(m_0, m_1, p_{93}, (H^e - Q_{93}, H^d - Q_{92}, H^c - Q_{91}))$s which are computed in the last step, where $p_{93} = (Q_{86} \| Q_{87} \| Q_{88} \| Q_{89} \| Q_{90} \| Q_{91} \| Q_{92} \| Q_{93})$.
5. For all (m_5, m_{11}),
   (a) do the following: $p_j \leftarrow R_j^{-1}(p_{j+1}, m_{\pi(j)})$ for $j = 39, 38, \ldots, 2$, where, $p_2 = (Q_{-5} \| Q_{-4} \| Q_{-3} \| Q_{-2} \| Q_{-1} \| Q_0 \| Q_1 \| Q_2)$.
   (b) Check whether Q_{-5}, Q_{-4}, and Q_{-3} are matched with $H^c - Q_{91}, H^d - Q_{92}$, and $H^e - Q_{93}$ in the table.
   (c) If they are matched, compute $p_{94}, p_{95}, p_{96}, p_0$, and p_1 by using the matched pairs, and check whether $H_n = p_0 + p_{96}$ are satisfied.
   (d) If satisfied, the pair of corresponding message and p_0 is a pseudo-preimage of H_n.

Table 6. Message word distribution for 4-pass HAVAL

Step	0	1	2	3	4	5	6	7	···	20	21	22	23	24	25	26	27	28	29	30	31
index	0	1	2	3	4	⑤	6	7	···	20	21	22	23	㉔	25	26	27	28	29	30	31
	second chunk													local collision (1-cycle)							
Step	32	33	34	···	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	
index	⑤	14	26	···	30	3	21	9	17	㉔	29	6	19	12	15	13	2	25	31	27	
		first chunk																			
Step	64	65	66	67	68	69	70	71	72	73	74	75	76	77	···	90	91	92	93	94	95
index	19	9	4	20	28	17	8	22	29	14	25	12	㉔	30	···	21	10	23	11	⑤	2
	first chunk																			skip	
Step	96	97	98	···	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	
index	㉔	4	0	···	22	11	31	21	8	27	12	9	1	29	⑤	15	17	10	16	13	
		second chunk																			

In the above procedure, the complexity of step 3 is $2^{64} \cdot \frac{53}{96}$ and the complexity of step 5a is $2^{64} \cdot \frac{38}{96}$. After step 5b, $2^{32}(= 2^{128} \cdot 2^{-96})$ pairs are expected to be remained. After step 5c, $2^{-128}(= 2^{-160} \cdot 2^{32})$ pair are expected to be remained. Therefore, by repeating the above procedure 2^{128} times, we expect to obtain a pseudo-preimage, where the complexity is $2^{192}(= 2^{64} \cdot 2^{128})$. Finally, this pseudo-primage attack is converted to a preimage attack of the complexity of 2^{225} by the generic approach explained in Section 2.2[4]. Step 4 requires 13×2^{64} words of memory and other steps require negligible amount of memory.

5.2 A preimage attack on 4-pass HAVAL

We propose a preimage attack on 4-pass HAVAL, which finds a pseudo-preimage of 4-pass HAVAL at the complexity of 2^{224}, and is converted to a preimage attack of the complexity of 2^{241}. Thus, the resulting preimage is 2-block long. This attack uses the splice-and-cut, partial-matching, and local-collision techniques as shown in Table 6.

In this attack, we need to guarantee that the neutral words form a local-collision in Steps 24-32. This is achieved by fixing chaining variables so that the change of a chaining variable corresponding to both neutral words does not propagate through the Boolean functions. How chaining variables are fixed is shown in Table 7, where, $\mathbf{0}$, $\mathbf{1}$, $\mathbf{C}_i$, and $*$ denote `0x00000000`, `0xffffffff`, a fixed value, and a flexible value which depends on the value of neutral words, respectively.

The attack procedure for a hash value $H_n = (H^a \| H^b \| H^c \| H^d \| H^e \| H^f \| H^g \| H^h)$ is as follows.

Attack procedure

1. Randomly choose the values of $\mathbf{C}_1, \ldots, \mathbf{C}_5$, and fix the values of chaining variables denoted by $\mathbf{C}_1, \ldots, \mathbf{C}_5, \mathbf{0}$, and $\mathbf{1}$ in Table 7.
2. Compute m_i $(i \in \{25, 26, 27, 28\})$ by solving the step function.

[4] Combination of the attack proposed by Aumasson et al. described in Section 2.3 and P^{3}graph proposed in [3] will be the preimage attack with a complexity of 2^{225}. Moreover, following [1, Appendix], the complexity is further improved to 2^{224}, but the length of the preimage message will be very long.

Table 7. Fixed values for preimage attack on 4-pass HAVAL

step j	$m_{\pi(j)}$	Q_{j-7}	Q_{j-6}	Q_{j-5}	Q_{j-4}	Q_{j-3}	Q_{j-2}	Q_{j-1}	Q_j
24	$\textcircled{m}_{24}$	$\textcircled{Q}_{17}$	$\mathbf{C}_1$	$\mathbf{C}_2$	$\mathbf{C}_3$	$\mathbf{C}_4$	**1**	**0**	**0**
25	m_{25}	$\mathbf{C}_1$	$\mathbf{C}_2$	$\mathbf{C}_3$	$\mathbf{C}_4$	**1**	**0**	**0**	$*$
26	m_{26}	$\mathbf{C}_2$	$\mathbf{C}_3$	$\mathbf{C}_4$	**1**	**0**	**0**	$*$	**0**
27	m_{27}	$\mathbf{C}_3$	$\mathbf{C}_4$	**1**	**0**	**0**	$*$	**0**	**0**
28	m_{28}	$\mathbf{C}_4$	**1**	**0**	**0**	$*$	**0**	**0**	**0**
29	$\underline{m_{29}}$	**1**	**0**	**0**	$*$	**0**	**0**	**0**	$\mathbf{C}_5$
30	$\underline{m_{30}}$	**0**	**0**	$*$	**0**	**0**	**0**	$\mathbf{C}_5$	$\mathbf{C}_6$
31	$\underline{m_{31}}$	**0**	$*$	**0**	**0**	**0**	$\mathbf{C}_5$	$\mathbf{C}_6$	$\mathbf{C}_7$
32	$\textcircled{m}_5$	$*$	**0**	**0**	**0**	$\mathbf{C}_5$	$\mathbf{C}_6$	$\mathbf{C}_7$	$\mathbf{C}_8$
33		**0**	**0**	**0**	$\mathbf{C}_5$	$\mathbf{C}_6$	$\mathbf{C}_7$	$\mathbf{C}_8$	$\textcircled{Q}_{33}$

Messages used for the padding string are underlined.
Variables which we try all possible values are circled.

3. Fix m_{29}, m_{30}, and m_{31} to satisfy the padding for a 2-block message.
4. Compute Q_{30}, Q_{31}, and Q_{32} by the step function.
5. Randomly determine other message words that are not specified yet.
6. For all (m_5, Q_{17}), do the following:

$$\begin{cases} p_j \quad \leftarrow R_j^{-1}(p_{j+1}, m_{\pi(j)}) & \text{for } j = 23, 22, \ldots, 0, \\ p_{128} \leftarrow H_n - p_0, & \\ p_j \quad \leftarrow R_j^{-1}(p_{j+1}, m_{\pi(j)}) & \text{for } j = 127, 126, \ldots, 97. \end{cases}$$

7. Make a table of (m_5, Q_{17}, p_{97})s which are computed in the last step, where $p_{97} = (Q_{90} \| Q_{91} \| Q_{92} \| Q_{93} \| Q_{94} \| Q_{95} \| Q_{96} \| Q_{97})$.
8. For all (m_{24}, Q_{33}),
   (a) do the following: $p_{j+1} \leftarrow R_j(p_j, m_{\pi(j)})$ for $j = 33, 34, \ldots, 93$, where, $p_{94} = (Q_{87} \| Q_{88} \| Q_{89} \| Q_{90} \| Q_{91} \| Q_{92} \| Q_{93} \| Q_{94})$.
   (b) Check whether $Q_{94}, Q_{93}, Q_{92}, Q_{91}$, and Q_{90} are matched with those stored in the table.
   (c) If they are matched, compute p_{95}, p_{96}, and p_{97} with the matched pairs, and check whether they are matched with those stored in the table.
   (d) If matched, compute Q_{25}, which is denoted by $*$ in Table 7, by the step function for Step 24 with matched (m_{24}, Q_{17}) and by the step function for Step 33 with matched (m_5, Q_{33}).
   (e) Check whether both results of Q_{25} are matched.
   (f) If matched, the pair of corresponding message and p_0 is a pseudo-preimage of H_n.

In the above procedure, the complexity of step 6 is $2^{64} \cdot \frac{55}{128}$ and the complexity of step 8a is $2^{64} \cdot \frac{61}{128}$. After step 8b, $2^{-32} (= 2^{128} \cdot 2^{-160})$ pair is expected to be remained. After step 8c, $2^{-128} (= 2^{-32} \cdot 2^{-96})$ pair is expected to be remained. After step 8e, $2^{-160} (= 2^{-32} \cdot 2^{-128})$ pair is expected to be remained. Therefore, by repeating the above procedure 2^{160} times, we expect to obtain a pseudo-preimage, where the complexity is $2^{224} (= 2^{64} \cdot 2^{160})$. Finally, this pseudo-primage attack is converted to a preimage attack of the complexity of 2^{241} by the generic approach explained in Section 2.2. Step 7 requires 10×2^{64} words of memory and other steps require negligible amount of memory.

Table 8. Message word distribution for 5-pass HAVAL (full)

Step	0	1	2	3	4	5	···	19	20	21	22	23	24	25	26	27	28	29	30	31
index	0	1	2	3	4	5	···	19	⑳	21	22	23	24	25	㉖	27	28	29	30	31
	second chunk														skip					
Step	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	···	63
index	5	14	㉖	18	11	28	7	16	0	23	⑳	22	1	10	4	8	30	3	···	27
	skip																			
Step	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	···	95
index	19	9	4	⑳	28	17	8	22	29	14	25	12	24	30	16	㉖	31	15	···	2
	skip				first chunk															
Step	96	···	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	···	121	···
index	24	···	23	㉖	6	30	⑳	18	25	19	3	22	11	31	21	8	27	···	29	···
	first chunk						local collision (3-cycle)													
Step	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	···	159
index	27	3	21	㉖	17	11	⑳	29	19	0	12	7	13	8	31	10	5	9	···	15
	local collision				second chunk															

5.3 Notes on preimage attack on 5-pass HAVAL

A preimage attack on 5-pass HAVAL reduced to 151 steps

5-pass HAVAL reduced to 151 steps, which use the first 151 steps of 5-pass HAVAL, can be attacked by using the almost same approach as the attack on 4-pass HAVAL. In Table 6, Step 127 is a part of the second chunk that includes m_5 and is independent of m_{24}. According to the message expansion shown in Table 2, Steps 128-150 are independent from m_{24}. Therefore, the attack on 4-pass HAVAL in the last section can also be applied to the first 151 steps of 5-pass HAVAL. The complexity is almost the same, so we can find a pseudo-preimage at the complexity of 2^{224}, and this attack is converted to a preimage attack of the complexity of 2^{241}, and requires 10×2^{64} words of memory. Note, we experimentally confirmed that there is no selection of chunks that can attack more than 151 steps at the better complexity.

A preimage attack on full 5-pass HAVAL

As mentioned in Section 3.2, our attack works efficiently on a hash function with less than or equal to 4 rounds, but does not work on the one with more than 4 rounds. However, by combining the exhaustive search, we can find a pseudo-preimage at $2^{253.81}$.

To attack full 5-pass HAVAL, we need to use all the techniques explained: splice-and-cut, partial-matching, and local-collision techniques. The selection of the chunks are shown in Table 8. We stress that our computer search program did not find a pair of chunks that can be attacked with a 9-step local collision. This problem was solved by using a long collision path introduced in Section 3.2.

To guarantee that the neutral words form a local-collision in Steps 107-131, we fix chaining variables as shown in Table 9.

1. Fix the value of chaining variables as shown in Table 9, and derive the corresponding messages by using the step function.
2. Fix the value of message words that are not used inside the local collision steps. Note there is enough message space to find a pseudo-preimage.

Table 9. Fixed values for preimage attack on 5-pass HAVAL

Round	Step j	$m_{\pi(j)}$	Q_{j-7}	Q_{j-6}	Q_{j-5}	Q_{j-4}	Q_{j-3}	Q_{j-2}	Q_{j-1}	Q_j
4R	107	$\textcircled{m}_{20}$	$\textcircled{Q}_{100}$	$\mathbf{C}_1$	$\mathbf{C}_2$	$\mathbf{C}_3$	**1**	**0**	**1**	**1**
	108	m_{18}	$\mathbf{C}_1$	$\mathbf{C}_2$	$\mathbf{C}_3$	**1**	**0**	**1**	**1**	$*(Q_{108})$
	109	m_{25}	$\mathbf{C}_2$	$\mathbf{C}_3$	**1**	**0**	**1**	**1**	$*$	**1**
	110	m_{19}	$\mathbf{C}_3$	**1**	**0**	**1**	**1**	$*$	**1**	**1**
	111	$\circ m_3$	**1**	**0**	**1**	**1**	$*$	**1**	**1**	**1**
	112	m_{22}	**0**	**1**	**1**	$*$	**1**	**1**	**1**	**1**
	113	m_{11}	**1**	**1**	$*$	**1**	**1**	**1**	**1**	**0**
	114	m_{31}	**1**	$*$	**1**	**1**	**1**	**1**	**0**	**1**
	115	$(\circ m_{21})$	$*$	**1**	**1**	**1**	**1**	**0**	**1**	**1**
	116	m_8	**1**	**1**	**1**	**1**	**0**	**1**	**1**	$*(Q_{116})$
	117	$\circ m_{27}$	**1**	**1**	**1**	**0**	**1**	**1**	$*$	**1**
	118	m_{12}	**1**	**1**	**0**	**1**	**1**	$*$	**1**	**1**
	119	m_9	**1**	**0**	**1**	**1**	$*$	**1**	**1**	**1**
	120	m_1	**0**	**1**	**1**	$*$	**1**	**1**	**1**	**1**
	121	$\underline{m_{29}}$	**1**	**1**	$*$	**1**	**1**	**1**	**1**	**0**
	122	m_5	**1**	$*$	**1**	**1**	**1**	**1**	**0**	**0**
	123	(m_{15})	$*$	**1**	**1**	**1**	**1**	**0**	**0**	**0**
	124	m_{17}	**1**	**1**	**1**	**1**	**0**	**0**	**0**	$*(Q_{124})$
	125	m_{10}	**1**	**1**	**1**	**0**	**0**	**0**	$*$	**0**
	126	m_{16}	**1**	**1**	**0**	**0**	**0**	$*$	**0**	**0**
	127	m_{13}	**1**	**0**	**0**	**0**	$*$	**0**	**0**	$\mathbf{C}_4$
5R	128	$\circ m_{27}$	**0**	**0**	**0**	$*$	**0**	**0**	$\mathbf{C}_4$	**0**
	129	$\circ m_3$	**0**	**0**	$*$	**0**	**0**	$\mathbf{C}_4$	**0**	$\mathbf{C}_5$
	130	$\circ m_{21}$	**0**	$*$	**0**	**0**	$\mathbf{C}_4$	**0**	$\mathbf{C}_5$	$\mathbf{C}_6$
	131	$\textcircled{m}_{26}$	$*$	**0**	**0**	$\mathbf{C}_4$	**0**	$\mathbf{C}_5$	$\mathbf{C}_6$	$\mathbf{C}_7$
	132		**0**	**0**	$\mathbf{C}_4$	**0**	$\mathbf{C}_5$	$\mathbf{C}_6$	$\mathbf{C}_7$	$\textcircled{Q}_{132}$

Messages that appear twice are stressed with $\circ$.
Uninvolved messages are written in parentheses.

3. For all 2^{32} values of Q_{108}, compute a corresponding value of Q_{124}. Store the result in a table named Table A.
4. For all 2^{64} values of (m_{26}, Q_{100}), do the following:

$$p_j \leftarrow R_j^{-1}(p_{j+1}, m_{\pi(j)}) \qquad \text{for } j = 106, 105, \ldots, 68.$$

   Store $(m_{26}, Q_{100}, p_{68})$ in a table named Table B.
5. For all 2^{64} values of (m_{20}, Q_{132}), do the followings:

$$\begin{cases} p_{j+1} \leftarrow R_j(p_j, m_{\pi(j)}) & \text{for } j = 132, 133, \ldots, 159, \\ p_0 \quad\; \leftarrow H_n - p_{160}, \\ p_{j+1} \leftarrow R_j(p_j, m_{\pi(j)}) & \text{for } j = 0, 1, \ldots, 25. \end{cases}$$

   Store $(m_{20}, Q_{132}, p_{26})$ in a table named Table C.
6. For all 2^{96} values of $(m_{26}, Q_{100}, m_{20})$, do the followings.
   (a) Compute a value of Q_{108} by using (m_{20}, Q_{100}).
   (b) Find a value of corresponding Q_{124} by looking up Table A.
   (c) Compute a value of corresponding Q_{132} by using Q_{124} and m_{26}.

(d) Find values of corresponding p_{68} and p_{26} by looking up Table B and C.

(e) Compute skipped steps, which are Steps 26-67, by using $(m_{26}, p_{26}, m_{20}, p_{68})$.

(f) If skipped steps are matched, output corresponding messages.

In the above procedure, steps 1 and 2 finish in negligible time. Step 3 takes the complexity of about $2^{32} \cdot \frac{3}{160}$. Step 4 takes the complexity of $2^{64} \cdot \frac{39}{160}$, and step 5 takes the complexity of $2^{64} \cdot \frac{54}{160}$. Steps 6a to 6d finishes in negligible time for each of $(m_{26}, Q_{100}, m_{20})$. Step 6e seems to take the complexity of $2^{96} \cdot \frac{42}{160}$, but this can be easily improved to $2^{96} \cdot \frac{35}{160}$ by the partial-matching technique. Furthermore, the equation for computing Step 26 can be written as follows:

$$Q_{27} \leftarrow m_{\pi(26)} + (\text{term independent from } m_{\pi(26)}).$$

Therefore, Step 26 can be computed in negligible cost compared to one step function, and thus, the complexity becomes $2^{96} \cdot \frac{34}{160}$. After Step 6e, the number of matched message is evaluated as $2^{-160} (= 2^{-256} \cdot 2^{96})$. Therefore, by repeating steps 2 to 6 of the above procedure 2^{160} times, a pseudo-preimage can be found at the complexity of $2^{160} \cdot 2^{96} \cdot \frac{34}{160} \approx 2^{253.81}$. Steps 4 and 5 require 20×2^{64} words of memory in total and other steps require negligible amount of memory. To apply the depth first search for steps 4-6, Table B or C can be removed and memory requirement becomes half.

Notes on local collision shown in Table 9. In the local collision shown in Table 9, m_3, m_{21}, and m_{27} appear twice. Therefore, we need to be careful so that all fixed values in Table 9 can be achieved. m_{21} is used in Steps 115 and 130. Since a message used in Step 115 is an uninvolved message, we can determine m_{21} so that Step 130 is satisfied. We can ignore the influence to Step 115. Regarding m_3 and m_{27}, since they are used in Steps 129 and 128 whose outputs can be any value ($\mathbf{C}_6$ and $\mathbf{C}_5$), m_3 and m_{27} can be fixed so that Steps 111 and 117 are satisfied. This local collision also includes m_{29}, which is involved to the message padding. Unfortunately, this local collision needs to fix m_{29} to a unique value, since all input and output values of Step 121 are fixed. As a result, this attack cannot satisfy the message padding of 5-pass HAVAL. It is interesting that the uniquely fixed m_{29} satisfies the message padding rules of MD5. Since the padding rules of HAVAL require to produce more information than those of MD5, for example output length and pass number, the fixed m_{29} does not satisfy the padding for HAVAL but satisfies the padding for MD5.

6 Conclusion

In this paper, we proposed preimage attacks on HAVAL. We considered two general strategies to find a preimage. The first approach is speeding up the brute-force attack. By this approach, we can reduce the complexity of preimage attacks by a few bits. The second approach is the meet-in-the-middle approach. We found that the techniques proposed by [1] and [2] can be combined to attack hash functions with more rounds than previous works. As a result, we found a pseudo-preimage attack and a preimage attack on 3-pass HAVAL whose complexities are 2^{192} and 2^{225}, a pseudo-preimage attack and a preimage attack on 4-pass HAVAL whose complexities are 2^{224} and 2^{241}, and a pseudo-preimage attack and a preimage attack on 151-step 5-pass HAVAL whose complexities are also 2^{224} and 2^{241}. Moreover, we optimized the computational order for brute force attack on 5-pass HAVAL and its complexity is $2^{254.89}$. As far as we know, the proposed attack on 3-pass HAVAL is the best attack and proposed attacks on 4-pass HAVAL and 5-pass HAVAL are the first attacks.

References

1. Kazumaro Aoki and Yu Sasaki. Preimage attacks on one-block MD4, 63-step MD5 and more. In Roberto Avanzi, Liam Keliher, and Francesco Sica, editors, *Selected Areas in Cryptography — Workshop Records of 15th Annual International Workshop, SAC 2008*, pages 82–98, Sackville, New Brunswick, Canada, 2008.
2. Jean-Philippe Aumasson, Willi Meier, and Florian Mendel. Preimage attacks on 3-pass HAVAL and step-reduced MD5. In Roberto Avanzi, Liam Keliher, and Francesco Sica, editors, *Selected Areas in Cryptography — Workshop Records of 15th Annual International Workshop, SAC 2008*, pages 99–114, Sackville, New Brunswick, Canada, 2008. (also appeared in IACR Cryptology ePrint Archive: Report 2008/183 `http://eprint.iacr.org/2008/183`).
3. Christophe De Cannière and Christian Rechberger. Preimages for reduced SHA-0 and SHA-1. In David Wagner, editor, *Advances in Cryptology — CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 179–202, Berlin, Heidelberg, New York, 2008. Springer-Verlag. (slides on preliminary results were appeared at ESC 2008 seminar `http://wiki.uni.lu/esc/`).
4. Hans Dobbertin. The first two rounds of MD4 are not one-way. In Serge Vaudenay, editor, *Fast Software Encryption — 5th International Workshop, FSE'98*, volume 1372 of *Lecture Notes in Computer Science*, pages 284–292, Berlin, Heidelberg, New York, 1998. Springer-Verlag.
5. Jongsung Kim, Alex Biryukov, Bart Preneel, and Seokhie Hong. On the security of HMAC and NMAC based on HAVAL, MD4, MD5, SHA-0 and SHA-1. In Roberto De Prisco and Moti Yung, editors, *Security in Communication Networks SCN 2006*, volume 4116 of *Lecture Notes in Computer Science*, pages 242–256. Springer-Verlag, Berlin, Heidelberg, New York, 2006.
6. Jongsung Kim, Alex Biryukov, Bart Preneel, and Sangjin Lee. On the security of encryption modes of MD4, MD5 and HAVAL. In Sihan Qing, Wenbo Mao, Javier Lopez, and Guilin Wang, editors, *Information and Communications Security — 7th International Conference ICICS 2005*, volume 3783 of *Lecture Notes in Computer Science*, pages 147–158. Springer-Verlag, Berlin, Heidelberg, New York, 2005.
7. Eunjin Lee, Jongsung Kim, Donghoon Chang, Jaechul Sung, and Seokhie Hong. Second preimage attack on 3-pass HAVAL and partial key-recovery attacks on NMAC/HMAC-3-pass HAVAL. In Kaisa Nyberg, editor, *Fast Software Encryption — 15th International Workshop, FSE 2008*, volume 5086 of *Lecture Notes in Computer Science*, pages 189–206, Berlin, Heidelberg, New York, 2008. Springer-Verlag.
8. Gaëtan Leurent. MD4 is not one-way. In Kaisa Nyberg, editor, *Fast Software Encryption — 15th International Workshop, FSE 2008*, volume 5086 of *Lecture Notes in Computer Science*, pages 412–428, Berlin, Heidelberg, New York, 2008. Springer-Verlag.

9. Alfred John Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of applied cryptography.* CRC Press, 1997.
10. Ronald L. Rivest. *Request for Comments 1321: The MD5 Message Digest Algorithm.* The Internet Engineering Task Force, 1992. (`http://www.ietf.org/rfc/rfc1321.txt`).
11. Kazuhiro Suzuki and Kaoru Kurosawa. How to find many collisions of 3-pass HAVAL. In Atsuko Miyaji, Hiroaki Kikuchi, and Kai Rannenberg, editors, *Advances in Information and Computer Security, IWSEC 2007*, volume 4752 of *Lecture Notes in Computer Science*, pages 428–443, Berlin, Heidelberg, New York, 2007. Springer-Verlag. (A preliminary version was appeared in IACR Cryptology ePrint Archive: Report 2007/079 `http://eprint.iacr.org/2007/079`).
12. Bart van Rompay, Alex Biryukov, Bart Preneel, and Joos Vandewalle. Cryptanalysis of 3-pass HAVAL. In Chi Sung Laih, editor, *Advances in Cryptology — ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 228–245. Springer-Verlag, Berlin, Heidelberg, New York, 2003.
13. Xiaoyun Wang, Dengguo Feng, and Xiuyuan Yu. An attack on hash function HAVAL-128. *Science in China (Information Sciences)*, 48(5):545–556, 2005.
14. Xiaoyun Wang and Hongbo Yu. How to break MD5 and other hash functions. In Ronald Cramer, editor, *Advances in Cryptology — EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 19–35. Springer-Verlag, Berlin, Heidelberg, New York, 2005.
15. Zhangyi Wang, Huanguo Zhang, Zhongping Qin, and Qingshu Meng. Cryptanalysis of 4-pass HAVAL. (IACR Cryptology ePrint Archive: Report 2006/161 `http://eprint.iacr.org/2006/161`), 2006.
16. Hirotaka Yoshida, Alex Biryukov, Christophe De Cannière, Joseph Lano, and Bart Preneel. Non-randomness of the full 4 and 5-pass HAVAL. In Blundo Carlo and Cimato Stelvio, editors, *Security in Communication Networks SCN 2004*, volume 3352 of *Lecture Notes in Computer Science*, pages 324–336. Springer-Verlag, Berlin, Heidelberg, New York, 2004.
17. Hongbo Yu, Xiaoyun Wang, Aaram Yun, and Sangwoo Park. Cryptanalysis of the full HAVAL with 4 and 5 passes. In Matthew Robshaw, editor, *Fast Software Encryption — 13th International Workshop, FSE 2006*, volume 4047 of *Lecture Notes in Computer Science*, pages 89–110, Berlin, Heidelberg, New York, 2006. Springer-Verlag.
18. Yuliang Zheng, Josef Pieprzyk, and Jennifer Seberry. HAVAL — one-way hashing algorithm with variable length of output. In Jennifer Seberry and Yuliang Zheng, editors, *Advances in Cryptology — AUSCRYPT'92*, volume 718 of *Lecture Notes in Computer Science*, pages 83–104. Springer-Verlag, Berlin, Heidelberg, New York, 1993.